Vim Cheatsheet

Vim Cheatsheet

Vim (Vi IMproved) is a modal text editor. Every action is a keystroke — no menus, no mouse. Master the modes and you edit at the speed of thought.

Modes

```
# Start Vim
vim file.txt          # open or create file
vim                   # empty buffer
vim -R file.txt       # read-only mode
view file.txt         # alias for vim -R
```

Mode	Enter	Exit / Switch	Purpose
Normal	`Esc` (always returns here)	—	Navigation, commands, operators
Insert	`i` , `a` , `o` , `I` , `A` , `O`	`Esc`	Text input
Visual	`v` , `V` , `Ctrl+v`	`Esc`	Text selection
Command-line	`:`	`Enter` or `Esc`	Ex commands (save, quit, search)
Replace	`R`	`Esc`	Overtype characters
Select	`gh` (after `g`)	`Esc`	Like Visual but typing replaces selection

Navigation

Character & Line

```
h j k l       # left, down, up, right
0             # beginning of line
^             # first non-blank character
$             # end of line
gg            # first line of file
G             # last line of file
5gg or 5G     # go to line 5
```

Words, Sentences & Paragraphs

```
w             # next word start
b             # previous word start
e             # next word end
ge            # previous word end
W             # next WORD (whitespace-delimited)
B             # previous WORD
E             # end of WORD
)             # next sentence
(             # previous sentence
}             # next paragraph (blank-line separated)
{             # previous paragraph
%             # matching bracket/paren/brace
```

Scrolling

```
Ctrl+f        # page down
Ctrl+b        # page up
Ctrl+d        # half page down
Ctrl+u        # half page up
Ctrl+e        # scroll down one line
Ctrl+y        # scroll up one line
H             # top of screen
M             # middle of screen
L             # bottom of screen
zt            # scroll current line to top
zz            # scroll current line to center
zb            # scroll current line to bottom
```

Searching

```
/pattern      # search forward
?pattern      # search backward
n             # repeat search forward
N             # repeat search backward
*             # search forward for word under cursor
#             # search backward for word under cursor
:noh          # clear search highlight
```

Editing

Insert Mode Entry Points

```
i             # insert before cursor
a             # insert after cursor
I             # insert at beginning of line (before first non-blank)
A             # insert at end of line
o             # open line below
O             # open line above
s             # delete character and enter insert
S             # delete entire line and enter insert
```

Delete, Change, Yank

```
x             # delete character under cursor
X             # delete character before cursor
dd            # delete entire line
dw            # delete to next word start
d$ or D       # delete to end of line
d0            # delete to beginning of line
dgg           # delete to start of file
dG            # delete to end of file
cc or S       # change (delete + insert) entire line
cw            # change to next word start
c$ or C       # change to end of line
yy or Y       # yank (copy) entire line
yw            # yank word
y$            # yank to end of line
ygg           # yank to start of file
yG            # yank to end of file
p             # paste after cursor
P             # paste before cursor
```

Repeat & Undo

```
.             # repeat last edit command
u             # undo
Ctrl+r        # redo
```

Indentation

```
>>            # indent current line
<<            # unindent current line
5>>           # indent 5 lines
==            # auto-indent current line
gg=G          # auto-indent entire file
```

Search & Replace

```
# Basic substitute
:s/old/new/           # replace first occurrence on current line
:s/old/new/g          # replace all on current line
:s/old/new/gc         # replace all with confirmation

# Range substitute
:%s/old/new/g         # replace all in entire file
:5,20s/old/new/g      # replace in lines 5–20
:'<,'>s/old/new/g     # replace in visual selection

# Advanced patterns
:%s/\t/  /g           # replace tabs with 2 spaces
:%s/^\s\+//           # remove leading whitespace
:%s/\s\+$//           # remove trailing whitespace
:%s/foo\|bar/baz/g    # replace foo or bar with baz
```

Marks

```
m{a-z}        # set local mark (a-z)
m{A-Z}        # set global mark (file-wide, uppercase)
'{a-z}        # jump to local mark (line start)
'{A-Z}        # jump to global mark
''            # jump back to last jump position
`{a-z}        # jump to exact mark position (column)
:marks        # list all marks
:delmarks a b # delete marks a and b
```

Registers

```
"a            # use register 'a' for next delete/yank
"ayy          # yank line into register a
"ap           # paste from register a
"0p           # paste from last yank register
"+p           # paste from system clipboard
"+y           # yank into system clipboard
"*p           # paste from primary selection (X11)
:registers    # show all registers
```

Macros

```
q{a-z}        # start recording macro into register a
q             # stop recording
@a            # replay macro a
@@            # replay last used macro
5@a           # replay macro a five times
:reg a        # view contents of macro register a
```

Practical Macro Example

```
# Record a macro to wrap a word in quotes
qa             # start recording into register a
i"<Esc>        # insert opening quote, return to normal
ea             # move to end of word
a"<Esc>        # insert closing quote
q              # stop recording

# Apply to every word on a line
# Position cursor on first word, then:
qa i"<Esc> ea a"<Esc> q    # (this is the macro above)
100@a                       # replay 100 times (stops at end of line)
```

Visual Mode

```
v             # character-wise visual mode
V             # line-wise visual mode
Ctrl+v        # block visual mode
o             # go to other end of selection
gv            # re-select last visual selection
```

Visual Mode Operations

```
# After selecting text in visual mode:
d             # delete selection
c             # change selection
y             # yank (copy) selection
>             # indent selection
<             # unindent selection
~             # toggle case of selection
u             # lowercase selection
U             # uppercase selection
:sort         # sort selected lines
:!fmt         # reformat paragraph via fmt
```

Visual Block Mode

```
Ctrl+v        # enter block visual mode
I             # insert at left edge of block (applies to all lines)
A             # append at right edge of block
$             # extend block to end of each line
```

Practical Block Mode Example

```
# Comment out multiple lines with //
1. Move to first line
2. Ctrl+v to enter block mode
3. jj to select 3 lines down
4. I to insert at the block's left edge
5. Type // then Esc
6. The // is inserted on all selected lines
```

Buffers, Windows & Tabs

Buffers

```
:ls or :buffers     # list all buffers
:bnext or :bn       # next buffer
:bprev or :bp       # previous buffer
:bfirst             # first buffer
:blast              # last buffer
:b 3                # go to buffer 3
:b file.txt         # go to buffer by name (partial match)
:bd                 # close current buffer
:bd 3               # close buffer 3
:bw                 # wipe buffer (close + delete)
:e file2.txt        # open file in new buffer
:split file2.txt    # open file in horizontal split
:vsp file2.txt      # open file in vertical split
```

Windows (Splits)

```
:split or :sp       # horizontal split
:vsplit or :vsp     # vertical split
Ctrl+w s            # horizontal split
```

```
Ctrl+w v            # vertical split
Ctrl+w h            # move to left window
Ctrl+w j            # move to below window
Ctrl+w k            # move to above window
Ctrl+w l            # move to right window
Ctrl+w w            # cycle through windows
Ctrl+w W            # cycle backwards
Ctrl+w =            # equalize window sizes
Ctrl+w _            # maximize current window
Ctrl+w |            # maximize current window (vertical)
Ctrl+w q            # close current window
Ctrl+w o            # close all other windows
```

Tabs

```
:tabnew             # new tab
:tabedit file.txt   # open file in new tab
:tabclose or :tc    # close current tab
:tabnext or :tn     # next tab
:tabprev or :tp     # previous tab
:tabfirst or :tf    # first tab
:tablast or :tl     # last tab
:tabs               # list all tabs
gt                  # next tab
gT                  # previous tab
5gt                 # go to tab 5
```

File Operations

```
:w                   # save
:w!                  # force save (override permissions)
:w file.txt          # save as new file
:q                   # quit
:q!                  # quit without saving
:wq or ZZ            # save and quit
:x                   # save and quit (only if modified)
:e!                  # reload file from disk
:e file.txt          # open another file
:r file.txt          # insert contents of file below cursor
:r !command          # insert command output below cursor
:saveas file.txt     # save under new name
```

.vimrc Basics

```
# ~/.vimrc — essential settings

" General
set nocompatible              # required for Vim features
set encoding=utf-8
set fileencoding=utf-8
set number                    # show line numbers
set relativenumber            # relative line numbers for easy jumps
set scrolloff=8               # keep 8 lines of context when scrolling
set sidescrolloff=8
```

```
set signcolumn=yes            # always show sign column
set cursorline                # highlight current line
set colorcolumn=80            # highlight column 80
set wrap                      # wrap long lines
set linebreak                 # break at word boundaries
set showbreak=↪               # show wrap indicator
set breakindent               # indent wrapped lines

" Indentation
set tabstop=4
set shiftwidth=4
set softtabstop=4
set expandtab                 # use spaces instead of tabs
set smartindent
set autoindent
filetype indent on            # load indent rules per filetype

" Search
set hlsearch                  # highlight search results
set incsearch                 # incremental search
set ignorecase
set smartcase                 # case-sensitive when uppercase present

" Performance
set lazyredraw                " don't redraw during macros
set updatetime=300            " faster update time

" Persistent undo
set undofile
set undodir=~/.vim/undodir

" Key mappings
let mapleader=" "
nnoremap <leader>w :w<CR>
nnoremap <leader>q :q<CR>
nnoremap <C-h> <C-w>h
nnoremap <C-j> <C-w>j
nnoremap <C-k> <C-w>k
nnoremap <C-l> <C-w>l
```

Text Objects

Text objects are one of Vim's most powerful features. They allow operators (`d` , `c` , `y` , `v`) to act on grammatical units like words, sentences, and blocks.

```
# Inner vs Around
# i = inner (the content only, without delimiters)
# a = around (content + delimiters)

# Word
diw           # delete inner word (no surrounding whitespace)
daw           # delete around word (includes trailing whitespace)
ciw           # change inner word (delete + enter insert mode)
yiw           # yank inner word

# Quotes
di"           # delete text inside double quotes (not the quotes)
da"           # delete text inside quotes + the quotes themselves
```

```
ci'           # change text inside single quotes
ya`           # yank text inside backticks

# Parentheses, brackets, braces
di( or dib    # delete inside parentheses
da( or dab    # delete inside parentheses + parens
di{ or diB    # delete inside braces (B = block)
ci[           # change inside square brackets
da>           # delete inside angle brackets + brackets

# Tags (HTML/XML)
dit           # delete inside HTML/XML tags
dat           # delete around tags (content + opening/closing tags)
cit           # change inside tags

# Indent objects (Python, YAML, etc.)
dii           # delete inner indent (current indentation block)
dai           # delete around indent (including blank line above)
cii           # change inner indent block

# Paragraph objects
dip           # delete inner paragraph (text between blank lines)
dap           # delete around paragraph (including trailing blank line)
```

Command-Line Mode

```
# Run external commands
:!ls -la              # run ls and show output
:!make                # run make in the current directory
!!                    # replace current line with output of a command
!!ls                  # replace current line with ls output

# Read command output into buffer
:r !date              # insert current date below cursor
:r !curl -s https://api.ipify.org   # insert external data

# Sort and filter
:%sort                # sort entire file
:sort u               # sort and remove duplicates
:5,20sort             # sort lines 5-20
:'<,'>sort            # sort visual selection
:%!sort -rn           # pipe entire file through sort (reverse numeric)
:'<,'>!jq .           # pipe visual selection through jq

# Global command (execute on matching lines)
:g/foo/d              # delete all lines containing "foo"
:g/^$/d               # delete all blank lines
:g/foo/s/bar/baz/g    # replace bar with baz only on lines containing foo
:v/foo/d              # delete lines NOT containing "foo" (vglobal)

# Substitution flags
:%s/foo/bar/g         # replace all occurrences
:%s/foo/bar/gc        # replace all with confirmation
:%s/foo/bar/gI        # case-sensitive (override ignorecase)
:%s/foo/bar/gn        # report number of replacements without changing
:%s/foo/\=submatch(0) . " suffix"/g   # expression replacement

# Repeat last substitution
```

```
&                    # repeat last :s on current line
:g/pattern/s//new/g   # use last search pattern

# Other useful commands
:normal! gg=G         # auto-indent entire file via normal mode command
:put =getline(1,5)    # paste lines 1-5 below cursor
:delete               # delete current line
:move 0               # move current line to top of file
:copy $               # copy current line to bottom of file
```

Folding

```
# Fold methods
:set foldmethod=indent   # fold based on indentation (default, works well for most code)
:set foldmethod=syntax   # fold based on syntax rules (language-aware)
:set foldmethod=manual   # fold only where you manually create folds
:set foldmethod=marker   # fold at markers like {{{ and }}}

# Manual fold creation
zf3j                  # create fold: cursor + 3 lines down
zfa}                  # fold to matching brace
v...zf                # fold visual selection

# Fold navigation
za                    # toggle fold under cursor
zc                    # close fold
zo                    # open fold
zM                    # close all folds
zR                    # open all folds
zj                    # move to start of next fold
zk                    # move to end of previous fold
[z                    # move to start of current fold
]z                    # move to end of current fold

# Persistent folds (save across sessions)
:set foldmethod=manual
:mkview               # save folds
:loadview             # restore folds
# Add to .vimrc:
" au BufWinLeave ?* mkview
" au BufWinEnter ?* silent loadview
```

Search Settings

```
:set hlsearch          # highlight all search matches
:set nohlsearch        # disable highlight
:noh                   # temporarily clear highlight for current search

:set incsearch         # show matches as you type (incremental search)
:set noincsearch       # disable incremental

:set ignorecase        # case-insensitive search
:set smartcase         # override: case-sensitive when search contains uppercase

:set gdefault          # make the g flag default in :s (affects all on line)
:set wrapscan          # search wraps around end of file (default)
```

```
:set magic              # default: most metacharacters have special meaning
:set nomagic            # only ^ and $ are special

# Very magic mode (\v) — no need to escape most regex characters
/\vfoo(bar|baz)        # match foo(bar or foo(baz) without escaping parens
/\v\d{3}-\d{4}         # match phone number pattern
/\v^\s*\w+             # match lines starting with whitespace + word

# Practical search patterns
/\v<(foo|bar)>         # whole word match for foo or bar
/\v^[A-Z]              # lines starting with uppercase
/\v\s+$               # trailing whitespace
/\v^$\n^$             # consecutive blank lines
```

Advanced Substitution

```
# Sub-replace special characters
:%s/\n/ /g            # replace newlines with spaces (join lines)
:%s/\r//g             # remove carriage returns (Windows line endings)

# Expression replacement (\=)
# Number lines
:%s/^/\=line('.') . '. '   # prepend line numbers

# Increment numbers
:let i=1 | g/\d\+/s//\=i/ | let i+=1

# Submatch groups
:%s/\(\w\+\) (\(\w\+\))/\2: \1/g    # swap "first (last)" to "last: first"
:%s/\v(\w+) (\w+)/\2 \1/g           # same with very magic

# Range tricks
:.s/foo/bar/g          # current line only
:$s/foo/bar/g          # last line only
:%s/foo/bar/g          # entire file
:'<,'>s/foo/bar/g      # visual selection
:g/pattern/s/foo/bar/g # only on lines matching pattern
:.,+5s/foo/bar/g       # current line + 5 lines below

# The e flag (suppress "pattern not found" errors)
:%s/old_pattern/new/gc e

# Replace with line from register
:%s/pattern/\=@a/g     # replace with contents of register a
```

Abbreviations

```
# Insert-mode abbreviations (expand while typing)
:iabbrev @@ tobias@example.com
:iabbrev sig -- Tobias\n
:iabbrev dtdate \=strftime('%Y-%m-%d')

# Command-line abbreviations
:cabbrev W w
:cabbrev Q q
```

```
:cabbrev Wq wq
:cabbrev Qa qa

# View abbreviations
:abbrev

# Clear abbreviations
:iunabbrev @@
:cunabbrev W

# Use in .vimrc for common corrections
:iabbrev teh the
:iabbrev seperate separate
:iabbrev lenght length
```

External Commands & Filtering

```
# Run command and insert output
:r !date
:r !ls -la
:r !curl -s https://httpbin.org/ip
:r !grep -n "pattern" %

# Pipe buffer through external command
:%!python3 -m json.tool     # format JSON in entire file
:'<,'>!sort -u               # sort and dedupe selection
:'<,'>!column -t -s ','      # format CSV as table
:%!tr 'a-z' 'A-Z'            # uppercase entire file

# Visual selection + command
" Select text in visual mode, then:
:!fmt -w 80                 # reformat to 80 columns
:!python3 -c "import sys; print(sys.stdin.read().title())"

# Async commands
:!make &                    # run make in background
:terminal                   # open terminal in Vim buffer
```

Session Management

```
# Save and restore sessions
:mksession               # save session to Session.vim
:mksession!              # overwrite existing Session.vim
:mksession myproject.vim  # save to named file

# Restore sessions
:source Session.vim      # restore session
vim -S                   # restore default session
vim -S myproject.vim     # restore named session

# Session options (control what is saved)
:set sessionoptions=buffers,winsize,resize,winpos,curdir,folds

# Practical workflow
" In .vimrc:
```

```
" autocmd VimLeave * mksession! ~/.vim/last_session.vim
" Then restore with: vim -S ~/.vim/last_session.vim
```

Diff Mode

```
# Start diff mode
vimdiff file1.txt file2.txt       # open two files in diff mode
vim -d file1.txt file2.txt        # same as above

# Inside Vim
:diffthis                         # add current window to diff
:diffsplit otherfile.txt          # split and diff with another file
:diffpatch changes.patch          # diff current file against a patch

# Navigation in diff mode
]c                                # jump to next diff hunk
[c                                # jump to previous diff hunk
do                                # obtain (pull change from other file)
dp                                # put (push change to other file)
:diffupdate                       # re-scan for diffs after edits
:diffget                          # synonym for do
:diffput                          # synonym for dp

# Turn off diff for current window
:diffoff
```

Terminal Mode

```
# Open a terminal
:terminal                        # open terminal in horizontal split
:terminal bash                   # open specific shell
:10terminal                      # open with 10-line window

# Navigation in terminal mode
Ctrl-\ Ctrl-N                    # exit terminal mode (go to normal mode)
Ctrl-w N                         # same as above (exit terminal mode)

# Interact with terminal from normal mode
i                                # enter terminal insert mode
Ctrl-w "w                        # move to next window while in terminal mode

# Send commands to terminal
:term send ls -la\|head          # send command to terminal

# Useful patterns
:terminal make                   # run build in terminal
:terminal cargo test             # run tests in terminal
:terminal git log --oneline      # run git in terminal
```

Spell Checking

```
# Enable spell checking
:set spell                       # enable for current buffer
:set nospell                     # disable spell checking
```

```
:set spelllang=en_us            # set language (American English)
:set spelllang=de               # German
:set spelllang=en_us,de         # multiple languages

# Navigate spelling errors
]s                              # next spelling error
[s                              # previous spelling error
z=                              # show suggestions for word under cursor
1z=                             # use first suggestion
zg                              # add word under cursor to good word list
zw                              # add word under cursor to bad word list
zug                             # undo last zg/zw

# View spelling suggestions
:spellgood word                 # add word to dictionary
:spellbad word                  # mark word as incorrect
:spellsuggest                  # show suggestions for current word
```

Plugins & Plugin Managers

```
# --- vim-plug (minimal, fast plugin manager) ---
# Installation:
curl -fLo ~/.vim/autoload/plug.vim --create-dirs \
    https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim

# In .vimrc:
call plug#begin('~/.vim/plugged')

Plug 'junegunn/fzf', { 'do': { -> fzf#install() } }   " fuzzy finder
Plug 'junegunn/fzf.vim'                                " fzf integration
Plug 'preservim/nerdtree'                              " file explorer
Plug 'neoclide/coc.nvim', {'branch': 'release'}        " LSP autocomplete
Plug 'tpope/vim-fugitive'                              " git integration
Plug 'tpope/vim-commentary'                            " toggle comments
Plug 'vim-airline/vim-airline'                         " status bar

call plug#end()

# Commands:
:PlugInstall                   # install all plugins
:PlugUpdate                    # update plugins
:PlugClean                     # remove unused plugins

# --- lazy.nvim (modern Lua-based plugin manager) ---
# Requires Neovim 0.8+
# In init.lua:
-- local lazypath = vim.fn.stdpath("data") .. "/lazy/lazy.nvim"
-- vim.opt.rtp:prepend(lazypath)
```

```
# Essential plugin commands (vim-plug)
# FZF (fuzzy finder)
:Files                          " fuzzy find files
:Rg pattern                    " ripgrep search
:Buffers                       " fuzzy find open buffers
:Lines                          " fuzzy find lines in buffer
:Commits                       " fuzzy find git commits
```

```
# NERDTree
:NERDTree                       " toggle file explorer
:NERDTreeToggle                 " open/close NERDTree
:NERDTreeFind                   " find current file in tree

# Fugitive (git)
:Git                            " open git status
:Gdiffsplit                     " git diff split
:Gread                          " checkout file from git
:Gwrite                         " stage file
:Gblame                         " git blame

# Commentary
gcc                             " toggle comment on current line
gc                              " toggle comment in visual mode
```

Autocommands

```
# Define autocommands (use augroup to avoid duplication)
augroup my_autocommands
    autocmd!
    " Auto-remove trailing whitespace on save
    autocmd BufWritePre * %s/\s\+$//e

    " Return to last cursor position when opening a file
    autocmd BufReadPost *
        \ if line("'\"") > 0 && line("'\"") <= line("$") |
        \   exe "normal! g`\"" |
        \ endif

    " Set filetype-specific settings
    autocmd FileType python setlocal expandtab shiftwidth=4 tabstop=4
    autocmd FileType yaml setlocal expandtab shiftwidth=2 tabstop=2
    autocmd FileType go setlocal noexpandtab tabstop=4 shiftwidth=4

    " Auto-format on save (where applicable)
    autocmd BufWritePre *.go FormatWrite

    " Highlight column 80 for certain file types
    autocmd FileType python,rust setlocal colorcolumn=80

    " Set makeprg for different languages
    autocmd FileType python setlocal makeprg=python3\ -m\ py_compile\ %
    autocmd FileType go setlocal makeprg=go\ build\ .
augroup END
```

Recording & Advanced Macros

```
# Replay last command-line command
@:                              # repeat last :command
@@                              # repeat last macro

# Macro registers
"ap                             # play macro stored in register a
"b@                             # play macro stored in register b
```

```
# Recursive macros (macro that calls itself)
" Example: find and fix all occurrences of "foo" to "bar"
qa                              # start recording into register a
/foo<CR>                        # search for foo
cwbar<Esc>                      # change word to bar
@a                              # replay macro a (recursive!)
q                               # stop recording
100@a                           # replay 100 times (stops when search fails)

# Append to existing macro
qa                              # start recording into register a (overwrites)
q                               # stop recording (empty)
qaA                             # start APPENDING to register a
<edits>
q                               # stop recording

# Edit a macro as text
"ap                            # paste macro contents into buffer
" edit the pasted text
dd                             # delete the line into default register
"a                             # yank into register a (overwrites with edits)
```

Quickfix List

```
# Build system integration
:make                           " run make (or configured makeprg)
:make clean                     " run make clean
:cn                             " next error
:cp                             " previous error
:cfirst                         " first error
:clast                          " last error
:cc 3                           " jump to error number 3
:copen                          " open quickfix window
:cclose                         " close quickfix window
:cw                             " open quickfix if there are errors

# Grep integration
:grep "pattern" *.py            " grep in Python files
:grep -R "pattern" src/         " recursive grep
:Rg "pattern"                   " ripgrep (if installed, via fzf)

# Vimgrep (Vim's built-in grep)
:vimgrep /pattern/ *.py         " search for pattern in Python files
:vimgrep /pattern/j **/*.py     " search subdirectories (j = don't jump to first match)
:cn                             " next match
:copen                          " show all matches

# Location list (window-local quickfix)
:lopen                          " open location list
:lclose                         " close location list
:lnext                          " next location
:lprev                          " previous location
:ll                             " jump to current location

# Quickfix practical workflow
" Compile → fix errors → :cn → fix → :cn → fix → :ccl
```

Tips

Emergency: How to Exit

```
# If you're stuck in Vim, press Esc first, then:
:q!           # quit without saving
```

Recover from a Crash

```
vim -r file.txt       # recover from swap file
# After recovery:
# :write to save, then delete the swap file
```

Next Steps

- Bash CLI Tools Cheat Sheet — Unix command-line utilities for power users
- Git CLI Cheat Sheet — Version control commands and workflows
- Nano Cheat Sheet — Simple terminal editor for quick edits
- Emacs Cheat Sheet — Extensible Lisp-based editor